

AD-A040 611

SYRACUSE UNIV. N Y DEPT OF INDUSTRIAL ENGINEERING AND--ETC F/G 9/5
DUAL-MODE LOGIC FOR FUNCTION INDEPENDENT FAULT TESTING.(U)

MAY 77 S DASGUPTA, C R HARTMANN, L D RUDOLPH F30602-71-C-0312

UNCLASSIFIED

76-11

RADC-TR-77-151

NL

1 OF 1
ADA
040611



END

DATE
FILMED
7-77

AD A 040611

12

[Handwritten signature]

RADC-TR-77-151
Final Technical Report
May 1977



DUAL-MODE LOGIC FOR FUNCTION INDEPENDENT FAULT TESTING
Syracuse University

Approved for public release; distribution unlimited.

AD NO. _____
DDC FILE COPY

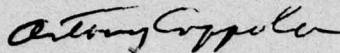
ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

DDC
JUN 14 1977
B

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public including foreign nations.

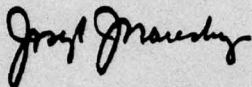
This report has been reviewed and is approved for publication.

APPROVED:



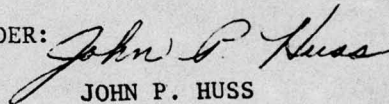
ANTHONY COPPOLA
Project Engineer

APPROVED:



JOSEPH J. NARESKY
Chief, Reliability & Compatibility Division

FOR THE COMMANDER:



JOHN P. HUSS
Acting Chief, Plans Office

Do not return this copy. Retain or destroy.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-77-151	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DUAL-MODE LOGIC FOR FUNCTION INDEPENDENT FAULT TESTING.	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report June 1976 - October 1976	6. PERFORMING ORG. REPORT NUMBER 76-11
7. AUTHOR(s) S. Dasgupta C. R. D. Hartmann L. D. Rudolph	8. CONTRACT OR GRANT NUMBER(s) F30602-71-C-0312	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45400526
9. PERFORMING ORGANIZATION NAME AND ADDRESS Syracuse University Department of Industrial Engineering Syracuse NY 13210	11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (RBRT) Griffiss AFB NY 13441	12. REPORT DATE May 1977
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same	13. NUMBER OF PAGES 33	15. SECURITY CLASS. (of this report) UNCLASSIFIED
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited. 16 4540 17 15		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same		
18. SUPPLEMENTARY NOTES RADC Project Engineer: Anthony Coppola (RBRT)		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Reliability Testing LSI Fault Detection Fault Testing		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A method of using hardware redundancy to ease the problem of fault testing in combinational and sequential logic circuits is presented. Dual-mode logic gates are used to construct combinational logic circuits which can be tested for all single stuck-at faults using just two function-independent tests. Analogous results for sequential circuits are also presented.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

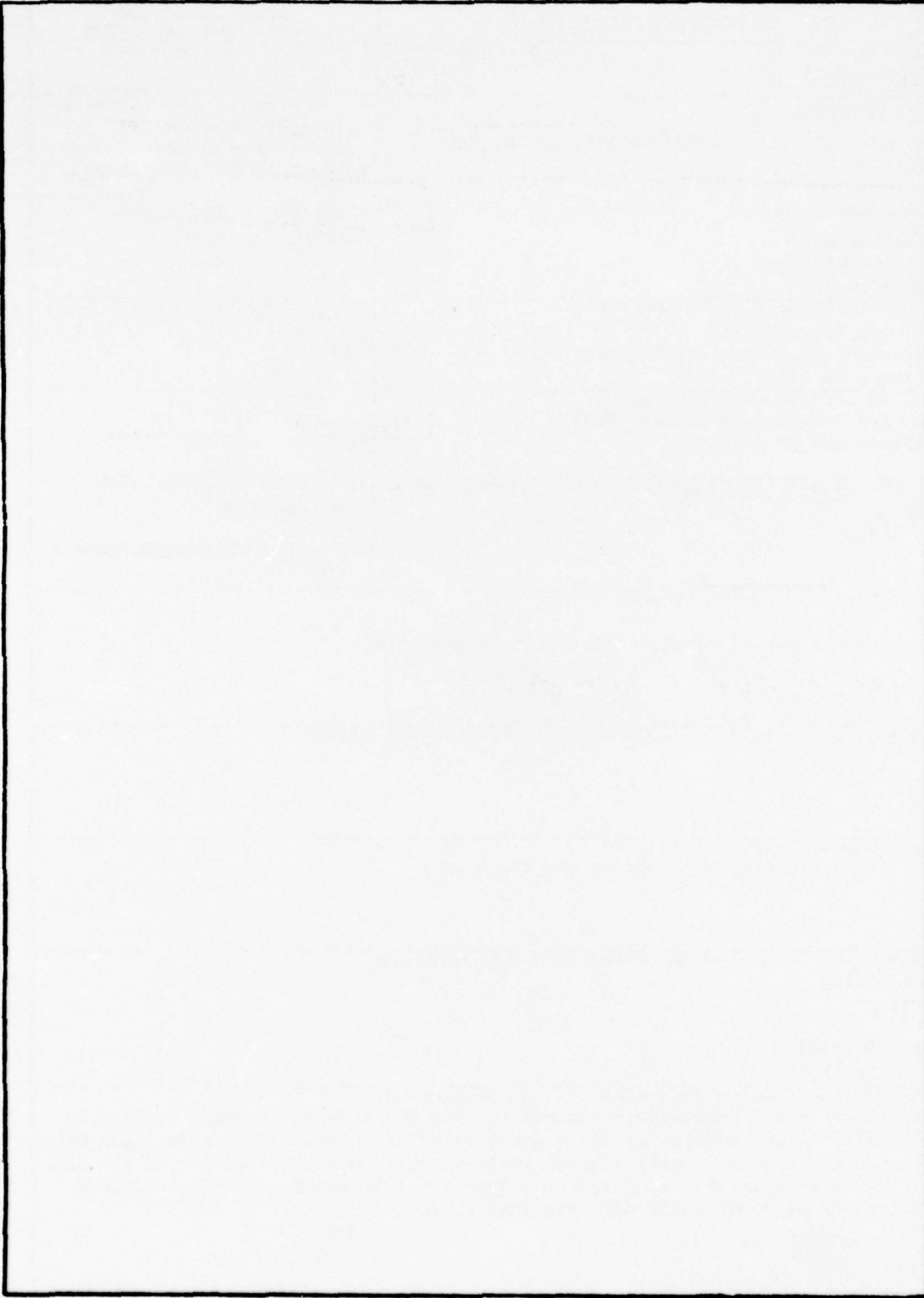
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409184

1/B

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

EVALUATION

This work presents simultaneously a possible major advance and a dangerous trap for the engineer looking for perfection in testing of LSI. There is no doubt that the dual-mode (2-T or 4-T) logic proposed here can be implemented, and would indeed test for all "stuck-at" faults (s-a) at the inputs and outputs of the 2-T or 4-T gates.

There would, for any particular logic family, be a penalty in performance. This is estimated to be at least a factor of four; a factor of about two in the components per logic function, and a doubling of the delay of each logic function. This would have to be balanced economically against the improved testability.

More seriously, a major assumption may be invalid, that is, an implementation can be found for which any internal node s-a fault will appear as a s-a fault at an input or output node. The 2-T gate, as proposed, is simply a gate with two extra inputs which convert it to an AND, OR, NAND, or NOR gate with respect to its normal inputs.

The obvious implementations of such logic all involve at least two stages, one of which is an inversion. This inversion itself might require a 2-T gate, etc., leading to an infinite number of components per logic function. It may therefore be impossible to create a valid implementation (i.e., no untestable internal nodes) with a finite number of components. A potential user must investigate this point considering that the gain required to renormalize digital signals can only be obtained with inversion.

Anthony Coppola

ANTHONY COPPOLA
Project Engineer

ACCESSION for	
NTIS	White Section <input checked="" type="checkbox"/>
DOC	Buff Section <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
Dist.	AVAIL. and/or SPECIAL
A	

1. INTRODUCTION

Traditionally, logic circuits have been designed with little regard for the problem of fault testing. Designers have been concerned with minimizing the complexity of the circuit and this was certainly understandable in the period prior to the development of LSI. When a logic circuit is small, there are a number of standard techniques for deriving tests to detect and locate faults [1,2]. As the size of the circuit grows, however, the number of tests required may increase very rapidly. Also, the complexity of finding the test set may become prohibitive. As logic component costs decrease and the difficulty of testing increases, a point is reached where logic designers should begin to consider the possibility of introducing hardware redundancy to simplify testing. In this report, we present one such approach based on the concept of dual-mode logic.

The basic idea is to use redundancy to (1) reduce the number of tests required, and (2) to reduce the complexity of deriving the tests. From a testing standpoint, the ultimate solution would be a method of designing logic networks that could be tested using a universal (i.e. function-independent) test set of minimum possible size. Although this might appear to be an unrealistic goal in general, we will show that just such a solution is possible for the restricted problem of detecting all single

stuck-at-faults (s-a-faults) in any combinational or sequential logic circuit. This is achieved through the use of a family of dual-mode universal logic gates. Two control inputs determine the mode of the gate. In the operating mode, the gate functions as a NAND, NOR, AND, OR or other desired logic element. In the test mode, all single s-a-faults on the inputs and output of the gate can be detected using just two test patterns. Furthermore, any combinational logic network built up from these gates will require only two control inputs and one test output, and can also be tested for all single s-a-faults using just two tests. The same approach can be used to design sequential logic circuits, in which case four extra inputs, one test output and six tests are required.

The family of dual-mode logic gates is introduced in Section II. Logic design of combinational and sequential circuits is considered in Sections III and IV, respectively, and a discussion of results is contained in Section V.

II. THE NEW LOGIC GATE

In this section we derive the logic function of the dual-mode logic gates. This gate forms a functionally complete set and can be tested for all single s-a-faults at its inputs and output using only two tests. Henceforth, this gate will be referred to as a "2-T gate".

Let X be a binary n -tuple and \bar{X} be the complement of X . The Hamming distance between any two binary n -tuples equals the number of positions in which they differ.

The necessary and sufficient conditions for the existence of a 2-T gate are:

- (i) there exists at least one input x for which the output is the complement of the output for the input \bar{x} .
- (ii) the output for any input n -tuple which is Hamming distance one away from x is the complement of the output for x .
- (iii) the output for any input n -tuple which is Hamming distance one away from \bar{x} is the complement of the output for \bar{x} .

This follows immediately from the fact that if a single fault exists in an input line of a 2-T gate, then application of x or \bar{x} will have the effect of applying an input n -tuple Hamming distance one away from x or \bar{x} . Thus, the output of the faulty gate will be the complement of the expected output. Since the output for x is different from that for \bar{x} , a fault in the output

will also be detected. Conversely, conditions (i), (ii) and (iii) must hold in order that x and \bar{x} form a complete test set for all single s-a-faults for a 2-T gate. Observe that a 2-T gate must not have exactly two inputs.

Table I shows a class of functions which satisfies the above conditions.

x_1	x_n	$f(x_1, x_2, \dots, x_n)$
0		...	0	0
1	0	...	0	1
0	1	0	...	1
.				.
.				.
0	0	...	0	1
1	1	0	...	x
.				.
.				.
0	0	1	...	\bar{x}
1	1	...	1	0
1	1	...	1	0
.				.
.				.
0	1	...	1	0
1	1	...	1	1

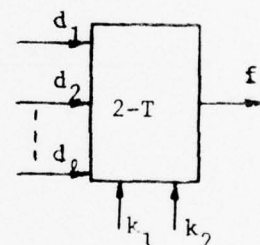
where x denotes "don't care".

TABLE I: Truth-table for a General n-input 2-T Gate.

Note that the two tests required to test for all single s-a-faults in the gate specified by Table I are $T_0 = (0, 0, \dots, 0)$ and $T_1 = (1, 1, \dots, 1)$.

We are now faced with the problem of specifying the values for the "don't care" conditions in Table I so that the resultant gate will form a functionally complete set. Table II shows one such assignment. (The input n-tuples in Table II are in standard lexicographical order).

CONTROL INPUTS		DATA INPUTS				$f(k_1, k_2, d_1, \dots, d_\ell)$
k_1	k_2	d_1	...	d_ℓ		
0	0	0	...	0	0	0
0	0	0	...	0	1	1
0	0	0	...	1	0	1
0	0	0	...	1	1	1
		\vdots			\vdots	
0	1	1	...	1	0	1
0	1	1	...	1	1	0
1	0	0	...	0	0	1
1	0	0	...	0	1	0
1	0	0	...	1	0	0
1	0	0	...	1	1	0
		\vdots			\vdots	
1	1	1	...	1	0	0
1	1	1	...	1	1	1



where $n = 1 + 2$.

TABLE II: Universal 2-T Gate

Observe that the gate described in Table II uses the same test set as before, i.e., T_0 and T_1 . Moreover, as Table III shows, the logic values assumed by the controls k_1 and k_2 determine the function of this gate. An important feature of this gate is:

Property 1:

The test set $\{T_0, T_1\}$ not only detects all single s-a-faults but also detects all possible combinations of 1 or fewer s-a-faults in the data inputs d_1, d_2, \dots, d , provided that the control inputs k_1, k_2 are fault-free.

k_1	k_2	(k_1, k_2, d)
0	0	OR(\underline{d})
0	1	NAND (\underline{d})
1	0	NOR (\underline{d})
1	1	AND (\underline{d})

TABLE III: Control Specifications for Logic Functions.

Thus far, we have described a new gate which can be tested for all single s-a-faults in general, and all combinations of s-a-faults confined to data inputs, using only two tests. Moreover, if the control inputs k_1 and k_2 are held at fixed values, then this gate will operate as a

universal gate (NAND or NOR). In the following sections we will use this 2-T gate as a building block for combinational and sequential networks and derive function-independent test sets for such networks.

III. COMBINATIONAL CIRCUITS

First, we will consider networks built with NAND gates. Since NAND gates are universal, this allows us to consider any general combinational network. Obviously, the most straightforward conversion to a 2-T network is to replace each NAND gate with a 2-T gate having the same number of data inputs. All k_1 inputs to these gates are then connected together and driven from a common package input, and similarly for the k_2 inputs. During normal operation k_1 is set to 0 and k_2 to 1. This method of conversion dictates that data inputs and control inputs be kept separate as shown in Figure 1.

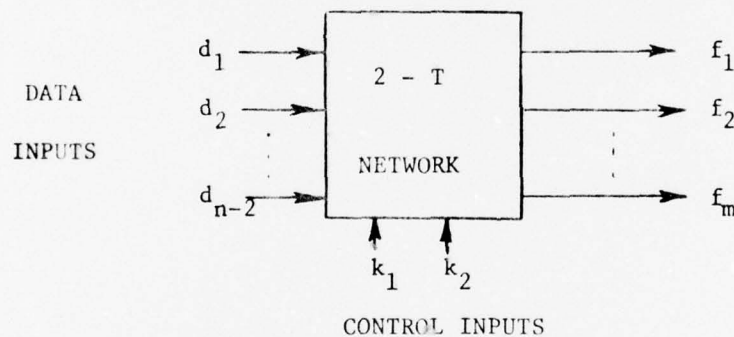


FIGURE 1 : n-input, m-output 2-T network.

The network of Figure 1 has the following characteristic:

Property 2

If all the inputs (data and control) are held at 0, then all gates in the network will have output 0 and consequently the final output of the

network will be 0. Similarly, if all inputs are held at 1, then all gates will have output 1 and the final output of the network will also be 1.

We now show that any distribution of s-a-faults in the data lines and outputs of gates will be detected by the tests $(k_1, k_2, d_1, \dots, d_{n-1}) = (0, 0, \dots, 0) = T_0$ and $(k_1, k_2, d_1, \dots, d_{n-1}) = (1, 1, 1, \dots, 1) = T_1$.

Obviously, any fault in an output of the network is detectable by the test set $\{T_0, T_1\}$. So, without loss of generality, we will consider that the outputs of the network are fault-free. Moreover, we will concern ourselves only with faults appearing at the inputs of 2-T gates since a fault that appears in the output of a gate is indistinguishable from the corresponding faults at the inputs of the gates fed by that output.

So, consider a s-a-fault for which there exists a fault-free path to at least one of the outputs of the network. There may be many faults in the network, but at least one of them must have such a path. Let us assume that this fault is a stuck-at-0 (s-a-0) fault at the input of a 2-T gate and that we have applied test T_1 . In the absence of any fault, the output of this gate and all other gates in the network would be 1 by Property 2. Hence, the network outputs would also be 1. But with the s-a-0 fault described above, the output of that gate would be 0 by Property 1. Then, since there exists at least one fault-free path from this fault to a network output, this fault will be propagated to that output because all gates in that path will have at least one zero at their

data inputs. Hence, that network output would be 0. Therefore test vector, T_1 , is a test for any set of faults (in data lines) that contains at least one s-a-0 fault which can be propagated to a network output. Analogously, test vector, T_0 , is a test for any set of faults (in data lines) that contains at least one stuck-at-1 (s-a-1) fault which can be propagated to a network output.

So far we have dealt with faults in data lines with the assumption that the control inputs are fault free. Now we show that any single s-a-fault in the control inputs to a 2-T network is also detectable by the test set, $\{T_0, T_1\}$.

Assume that the fault is a s-a-0 fault. Then when test vector T_1 is applied, by Property 2, the inputs to the faulty gate will be either

$$(k_1, k_2, d_1, \dots, d_{n-2}) = (0, 1, 1, \dots, 1)$$

or

$$(k_1, k_2, d_1, \dots, d_{n-2}) = (1, 0, 1, \dots, 1).$$

In either case, the output of this gate will be 0. From this point on, this fault appears as a failure in one or more data lines of 2-T gates which have no faults in their control inputs. Thus, as discussed before, this fault will be detected by the test vector, T_1 .

Analogously, any single s-a-1 fault in a 2-T network that occurs at one of the control inputs of a 2-T gate will be detected by the test vector, T_0 .

All k_1 inputs of 2-T gates are driven from a common input, and similarly for the k_2 inputs. Thus, a single fault at the output of either the k_1 driver or the k_2 driver will affect all the gates of a 2-T network. These faults will be referred to as catastrophic faults. Depending upon the network, the test set $\{T_0, T_1\}$ may or may not detect a catastrophic fault. Figure 2 shows an example in which catastrophic faults are not detectable by $\{T_0, T_1\}$, while Figure 3 shows an example in which they are.

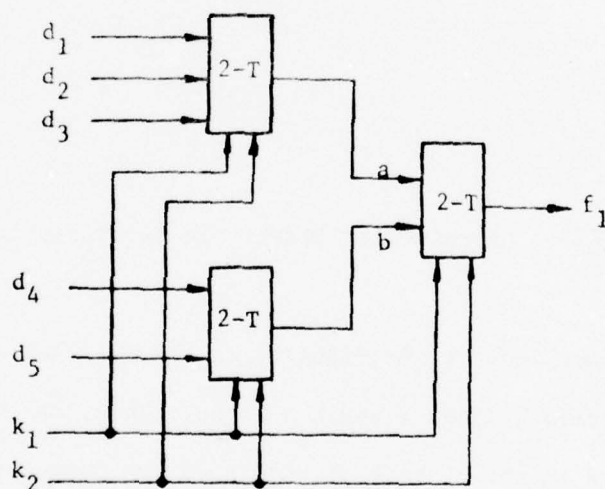


FIGURE 2: Circuit With Undetectable Catastrophic Faults.

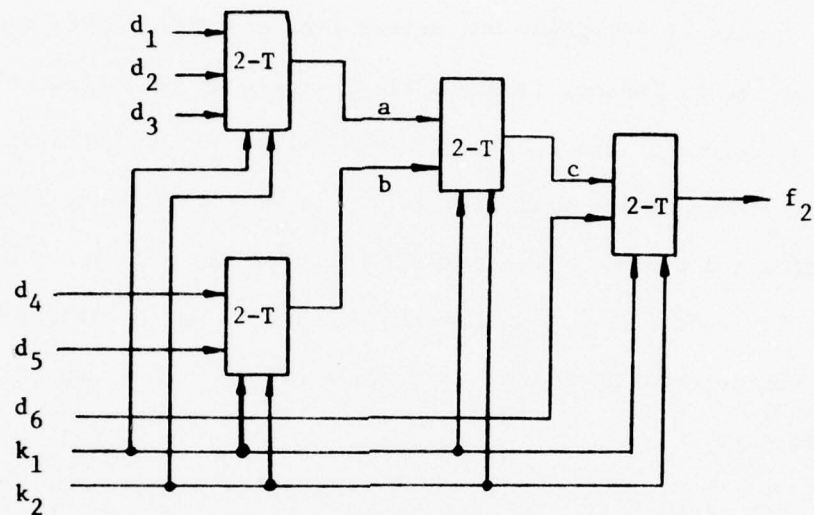


FIGURE 3: Circuit With Detectable Catastrophic Faults.

Suppose common input k_1 in Figure 2 is s-a-0. Then applying input vector T_1 will cause internal lines a and b to be 0. Thus, the final output f_1 will be 1. Since in the absence of the fault the output will also be 1, this fault will be undetectable. Consider the same fault in Figure 3. In this case, application of the input vector T_1 will cause internal lines a and b to be 0. Thus, internal line c will be 1 and since d_6 is also 1, f_2 will be 0. Hence, the fault is detectable. Similar analysis will

show that a s-a-1 fault in the common input k_1 and s-a-0 and s-a-1 faults in the common input k_2 are undetectable in Figure 2 but detectable in Figure 3.

In general, if a catastrophic fault in k_1 occurs, then the fault will be undetectable at a network output if the effect of the fault also appears at one or more data inputs of the gate which drives that output. On the other hand, if a catastrophic fault in k_2 occurs, then the fault will be undetectable at a network output only if the effect of the fault also appears at all the data inputs of the gate which drives that output. These two mechanisms of fault masking converge when the output gate has only a single data input, i.e., the output gate is an inverter in normal operation.

Now, if there exists any catastrophic fault which is not detectable at any of the network outputs, then the problem can be solved, as shown in Figure 4, by feeding one of the network outputs, say f_1 , to a gate with a single data input and observing its output at an extra network output, \hat{f}_1 . Since $(k_1, k_2) = (0,0)$ or $(1,1)$ implies test mode, a single catastrophic fault makes $(k_1, k_2) = (0,1)$ or $(1,0)$ the former being the condition for normal mode. The extra gate acts as an inverter when $(k_1, k_2) = (0,1)$ or $(1,0)$. So when a catastrophic fault is undetectable at any of the network

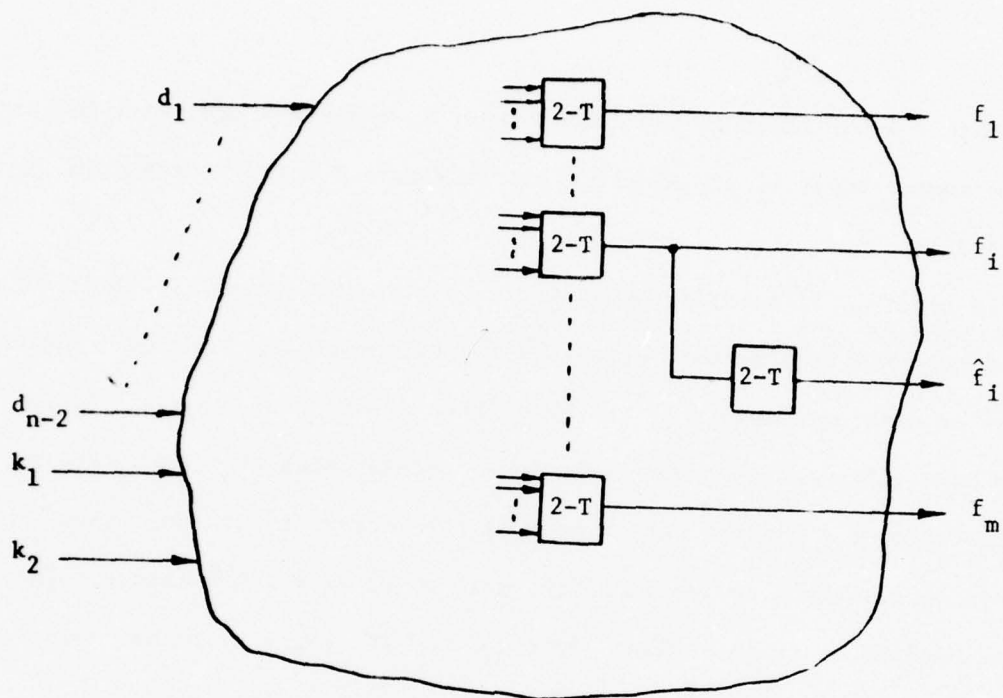


FIGURE 4: 2-T Network in Which Catastrophic Faults
in k_1 or k_2 are Detectable by T_0 or T_1 .

outputs, i.e. for any output f_i the logic value is the same with or without the existence of the fault, it is detectable at the output \hat{f}_i of the extra gate since its output is the complement of f_i for $(k_1, k_2) = (0,1)$ or $(1,0)$.

The designer has 2 options in implementing this solution. He can either use "feed forward" simulation to determine if the extra gate and network output is necessary or he can arbitrarily add the extra gate and

network output, knowing that whatever is undetectable at the regular network outputs will be detectable at the extra output.

In this section we have considered the 2-T combinational network. These networks require two extra inputs and possibly an extra output in order that all single s-a-faults be detected using the test set $\{T_0, T_1\}$. We next consider 2-T sequential networks.

IV. SEQUENTIAL CIRCUITS

In this section, we will present two methods of designing sequential circuits which can be tested for all single s-a-faults using a fixed, function-independent test set. In both methods, we assume that a fault is detectable if (1) it causes at least one of the outputs to be at a logic value which is the complement of the expected output, or (2) if it forces one or more of the outputs to oscillate, i.e., it switches its state at some frequency determined by the total delay in the faulty loop.

Consider a sequential network made with NAND gates. This network can be converted to a 2-T sequential network by replacing each NAND gate with a 2-T gate. The k_1 and k_2 control inputs of the gates are driven from common sources. As before, during normal operation, all k_1 inputs are set to 0 and all k_2 inputs are set to 1. In this 2-T sequential network, the first problem that arises is that of initializing the memory loops.

To understand this problem, consider the example of Figure 5. If

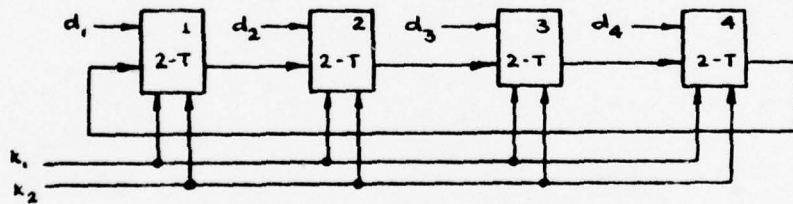


FIGURE 5: 2-T Sequential Network.

when T_0 is applied there is a 1 anywhere in the loop, the outputs of all the gates in the loop would be forced to 1, thus denoting the existence

of a fault even when one does not exist. A similar problem arises when T_1 is applied and there happens to be a 0 somewhere in the loop. Therefore, it must be possible to initialize the loop before a valid test can be performed.

Method 1:

Consider the 2-T gate of Figure 6 where the k_1 input is tied together with a data input to form a common input g_1 .

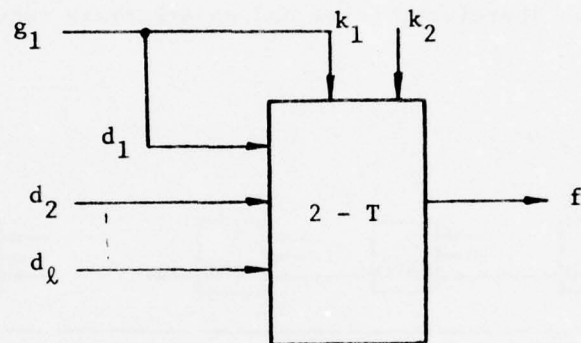


FIGURE 6: Specially Connected 2-T Gate.

Note that when $(g_1, k_2) = (0,1)$, the output of this gate is 1 independent of the other data inputs. Similarly, $(g_1, k_2) = (1,0)$ forces the output to 0. This gate could be used in a feedback memory loop to initialize the loop. There exists one minor problem with this configuration, however. This gate can never function as a NAND gate since that would require setting $(g_1, k_2) = (0,1)$ and hence setting $d_1 = 0$. Thus, the output of the gate will always be 1. Therefore, this gate can function as an AND gate when $(g_1, k_2) = (1,1)$ and as an OR gate when $(g_1, k_2) = (0,0)$. Thus, in the

first method of converting a NAND sequential network to a 2-T network, each memory loop in the latter network must have one gate connected as shown in Figure 6. The g_1 inputs in all loops are driven from a third common input. Since these gates act as AND gates in the operating mode, an extra gate has to be added to each memory loop which will act as an inverter. Therefore, if we had an arbitrary sequential network, as

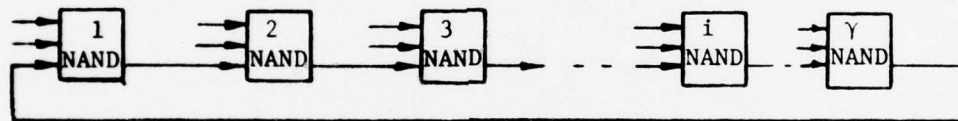


FIGURE 7: General Sequential Circuit.

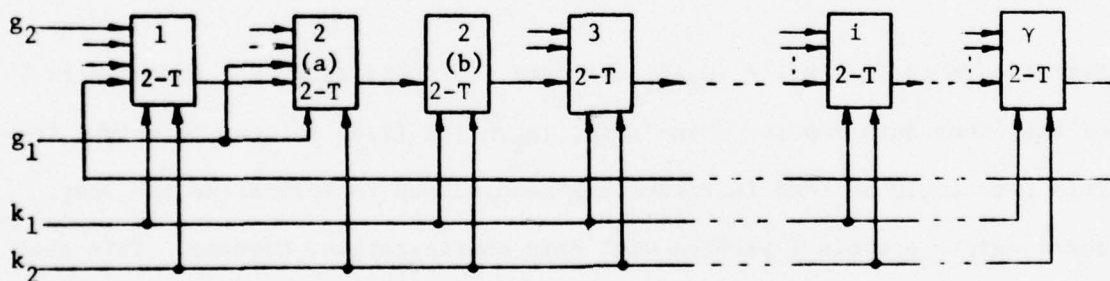


FIGURE 8: General 2-T Sequential Circuit (Method 1).

shown in Figure 7, its 2-T version using Method 1 would have the form of the network of Figure 8. Observe that gate 2(a) in Figure 8 has the special connection and 2(b) is the extra inverter. Note also that gate 1 has one extra data input, namely g_2 , which is shared with other memory loops and is used to test faults in g_1 and k_2 in gate 2(a).

One major drawback of this method is that all memory loops must have at least one of the outputs of gates 2(a) through r observable directly at a network output. This requirement is imposed because some faults may not be observable at the output of gate 1. Some designs might already have this property and would not be affected by this restriction.

We will also show that the following test set will detect all single s-a-faults, except catastrophic s-a-faults in the control inputs k_1 and k_2 , in any 2-T network which uses the initializing process of Method 1:

$$(k_1, k_2, g_1, g_2, d_1, \dots, d_{n-4}) = (0, 0, 1, 0, 0, \dots, 0) = I_0$$

$$(k_1, k_2, g_1, g_2, d_1, \dots, d_{n-4}) = (0, 0, 0, 0, 0, \dots, 0) = \hat{T}_0$$

$$(k_1, k_2, g_1, g_2, d_1, \dots, d_{n-4}) = (0, 0, 0, 1, 0, \dots, 0) = \hat{T}_1$$

$$(k_1, k_2, g_1, g_2, d_1, \dots, d_{n-4}) = (1, 1, 0, 1, 1, \dots, 1) = I_1$$

$$(k_1, k_2, g_1, g_2, d_1, \dots, d_{n-4}) = (1, 1, 1, 1, 1, \dots, 1) = \hat{T}_2$$

$$(k_1, k_2, g_1, g_2, g_1, \dots, d_{n-4}) = (1, 1, 1, 0, 1, \dots, 1) = \hat{T}_3.$$

Note that the subset $\{I_0, \hat{T}_0, \hat{T}_1\}$ represents an ordered sequence of tests as does the subset $\{I_1, \hat{T}_2, \hat{T}_3\}$.

Before we show that the above test set detects at least all single s-a-faults, except catastrophic faults which will be considered later, note that I_0 and I_1 initialize all loops in a fault-free network to 0 and 1 respectively, i.e., when I_0 and I_1 are applied, all gates in a loop will have 0 and 1 at their outputs respectively.

Consider the general network of Figure 9. It has three basic

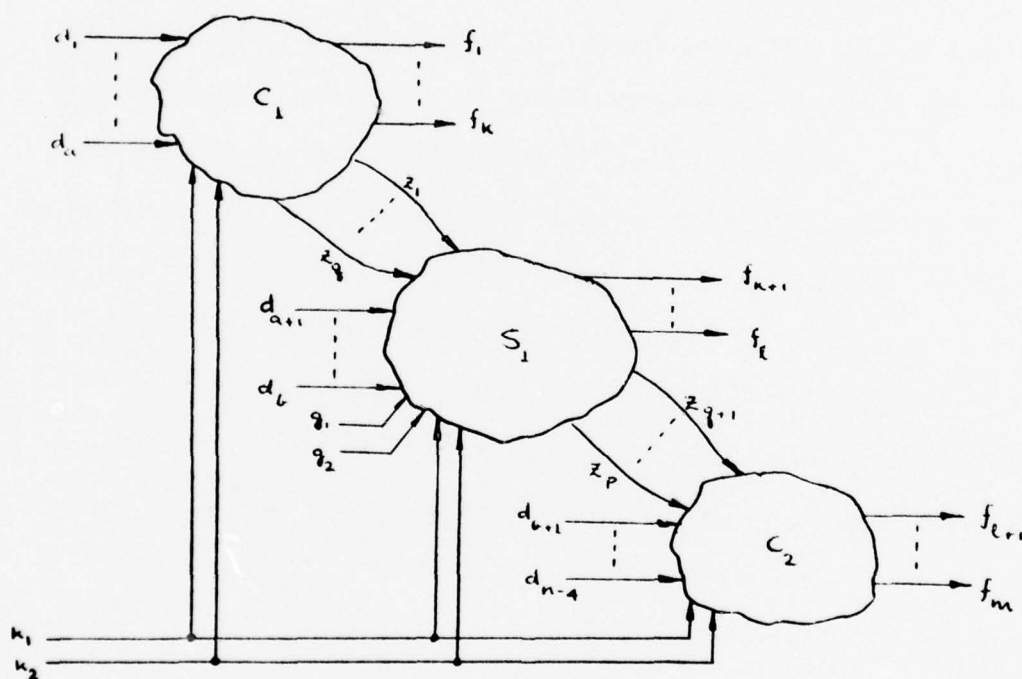


FIGURE 9; General 2-T Sequential Network (Method 1).

subsections. A combinational subnetwork C_1 feeds a sequential subnetwork S_1 . C_1 may also feed network outputs such as f_1 through f_k . S_1 may have some network inputs, such as d_{a+1} through d_b and it feeds some network outputs, some of which must be fed directly from the memory loops. It may also feed the combinational subnetwork C_2 . The latter feeds outputs f_{l+1} through f_m .

We will first show that any combination of faults in the data inputs and outputs of 2-T gates in the network of Figure 9 will be detected by the test set defined above. Subsequently, we will show that the same test set will detect all single s-a-faults in the control inputs of 2-T gates. As explained in Section III, we can assume, without loss of generality, that faults occur at data inputs of gates.

Consider the sequential network of Figure 9. Under fault-free conditions the expected output for \hat{T}_0 and \hat{T}_2 are $(f_1, \dots, f_m) = (0, \dots, 0)$ and $(f_1, \dots, f_m) = (1, \dots, 1)$ respectively. When T_1 is applied all those outputs from S_1 which are fed by the memory loops will be 1. Similarly, these same outputs will be 0 when \hat{T}_3 is applied.

Now consider any distribution of faults in the data inputs of 2-T gates. If one or more of these faults occur in C_2 , then as proven in Section III these faults are detectable either by \hat{T}_0 or \hat{T}_2 . The same argument shows that \hat{T}_0 or \hat{T}_2 will detect any faults in C_1 as long as one of them has a fault-free path to a network output. Furthermore, those faults

in C_1 that do not propagate to the network outputs will always propagate to S_1 when \hat{T}_0 or \hat{T}_2 is applied. And, since these output faults are indistinguishable from the corresponding input faults in S_1 , we need only show that distributed faults in S_1 are detectable.

Consider the subnetwork S_1 . It could contain blocks of memory loops, denoted by L , feeding each other directly or through combinational logic, denoted by C , as shown in Figure 10. Now as long as the control inputs k_1 , k_2 and g_1 are not malfunctioning, any faults internal to the combinational circuits in Figure 10 will be visible at the data inputs of the loops or at network outputs. So, to show that distributed faults in

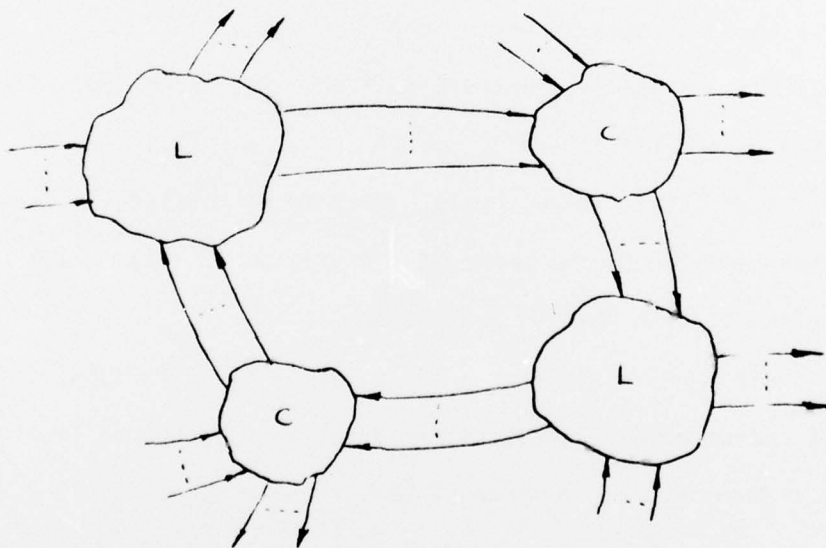


FIGURE 10: Subnetwork S_1 .

data inputs of gates in S_1 are detected we need only show that any set of data input faults in the loop are detectable by the test set defined earlier. Consider the general loop of Figure 8 in which each gate may have multiple data inputs, only one of which is fed by the gate preceding it in the loop. No restrictions are placed on the other data inputs. Then, for any set of faults there exists one fault which will be propagated to the network output fed by that loop when either \hat{T}_0 or \hat{T}_2 is applied.

Next, we consider single faults in control inputs of gates. A single control input fault in a gate in C_1 or C_2 in Figure 9 which has a path to a network output is detectable as shown in Section III. A similar fault in a gate in C_1 that appears at the data inputs of S_1 behaves like a data line fault in S_1 and hence is detectable. If a single control line fault occurs in the combinational circuit of Figure 10, then it either appears at the data input of the loops or at a network output. In either case it is a detectable fault.

Finally, we consider the loops in Figure 10. If there is a s-a-fault in the k_1 input of any of the gates in the loop except gate 2(a), then application of \hat{T}_0 or \hat{T}_2 will force the loop to oscillate. Hence, it is detectable. For any s-a-fault in the k_2 input of any of the same gates, application of \hat{T}_0 or \hat{T}_2 will either force the outputs of all the gates in the loop to assume an erroneous logic value or if the faulty gate has a single data input,

\hat{T}_0 or \hat{T}_2 will force the loop to oscillate. A fault in the k_2 input of gate 2(a) will be detected by \hat{T}_0 or \hat{T}_2 since it will force the outputs of every gate in the loop to assume an erroneous logic value. If the k_1 input fed by g_1 in gate 2(a) fails, then application of \hat{T}_0 or \hat{T}_2 will force the loop to oscillate. If the data input fed by g_1 fails, the failure behaves like any other data input fault and is detected by \hat{T}_0 or \hat{T}_2 . If input g_1 fails, then the fault is undetectable by \hat{T}_0 or \hat{T}_2 . But, it is detectable by \hat{T}_1 or \hat{T}_3 because under fault-free conditions, application of \hat{T}_1 or \hat{T}_3 forces the output of each gate in the loop to be 1 or 0 respectively. When g_1 fails, however, then application of \hat{T}_1 or \hat{T}_3 forces the output of every gate except gate 1 to be 0 or 1 respectively. Hence it is detectable. This latter fact also shows that the output of gate 1 does not always reflect the presence of a fault. Hence the required network output cannot be taken from gate 1.

Detection of a catastrophic fault in k_1 or k_2 is achieved in the same way as for combinational logic, i.e., an extra gate and an extra output may be necessary to ensure that all single catastrophic faults are detectable.

Method 2:

In this method, each memory loop is initialized by a special gate defined in Table IV. This gate has three extra inputs and is no longer testable for all single s-a-faults by \hat{T}_0 and \hat{T}_1 . For example if k_3 is

k_3	k_1	k_2	d_1	\dots	d_ℓ	f
0	0	0	0	\dots	0	0
0	0	0	0	\dots	0 1	1
				\vdots		\vdots
0	0	1	1	\dots	1 0	1
0	0	1	1	\dots	1	0
0	1	0	0	\dots	0	1
				\vdots		\vdots
0	1	1	1	\dots	1	1
1	0	0	0	\dots	0	0
				\vdots		\vdots
1	0	1	1	\dots	1	0
1	1	0	0	\dots	0	1
1	1	0	0	\dots	0 1	0
				\vdots		\vdots
1	1	1	1	\dots	1 0	0
1	1	1	1	\dots	1	1

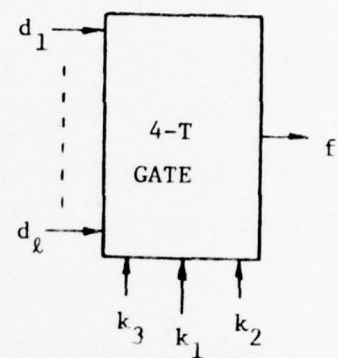


TABLE IV. Universal 4-T Gate.

s-a-0, then the test pattern $(k_3, k_1, k_2, d_1 \dots d_l) = (1, 1, 1, 1, \dots, 1)$ produces the same output for both fault-free and faulty operation. Hence, two more tests are required to test the two s-a-faults in k_3 . As Table IV shows any pattern with $(k_3, k_1, k_2) = (1, 0, 0)$ and a 1 in at least one data input of this gate is a test for k_3 s-a-0. Similarly, all patterns that are complements of the above tests are tests for k_3 s-a-1. Henceforth, this gate will be referred to as a 4-T gate.

In the second method of converting a NAND sequential network to a 2-T sequential network we replace one of the NAND gates in each loop with a 4-T gate while all other NAND gates are replaced by 2-T gates. All 2-T and 4-T gates have common k_1 and k_2 inputs while all 4-T gates have a common k_3 input. Figure 11 shows a general 2-T memory loop obtained by Method 2. To guarantee the universality of the two extra tests required to test the k_3 inputs it is necessary to reserve one of the data inputs of each 4-T gate as a gate input g_1 to be fed from a common network input, also called g_1 .

Consider the general sequential network of Figure 12. Analogously to Figure 9 it has two combinational subnetwork C_1 and C_2 . The intermediate subnetwork S_1 can be represented as in Figure 10. However, unlike Method 1 each loop in Figure 12 is not required to go to a separate network output.

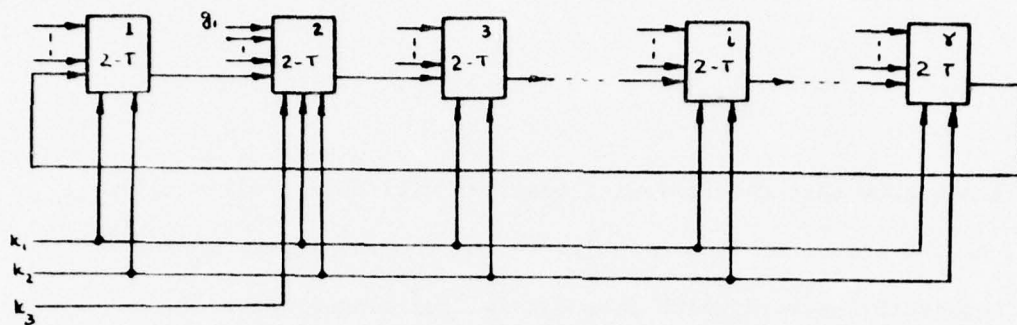


FIGURE 11: General 2-T Memory Loop (Method 2).

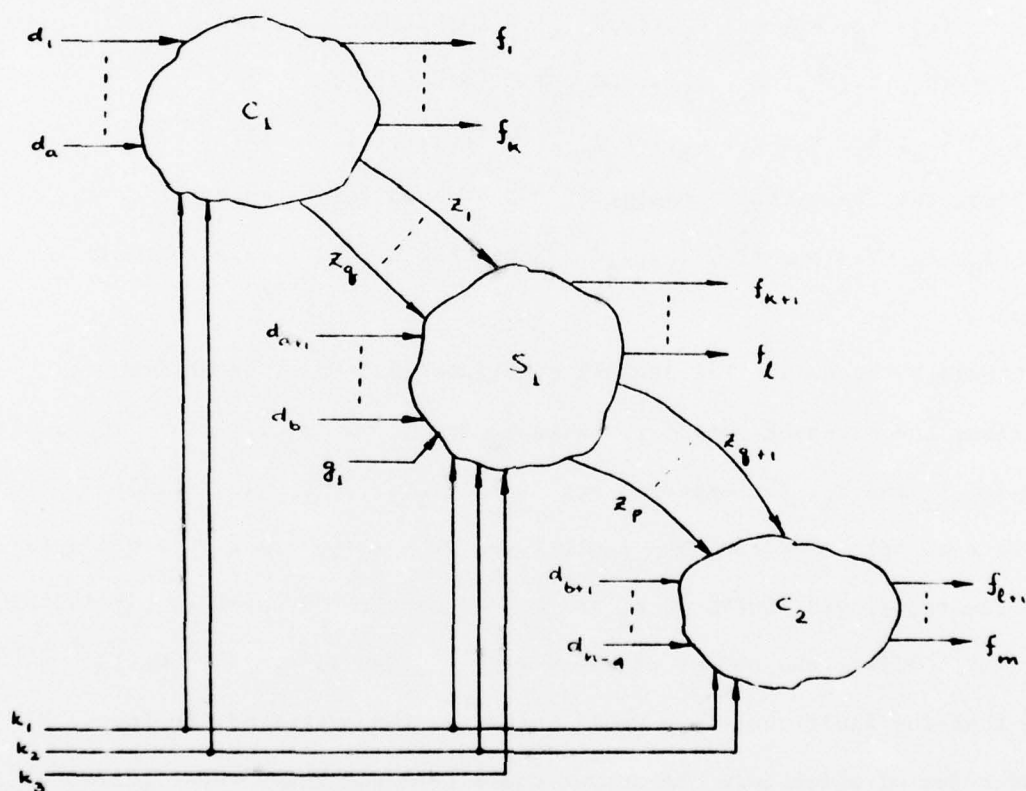


FIGURE 12: General 2-T Sequential Network (Method 2).

We will now show that the following test set will detect all single s-a-faults, except catastrophic s-a-faults in the control inputs k_1 and k_2 in any 2-T sequential network which uses the initializing process of

Method 2.

$$\hat{T}_0 = (k_3, k_1, k_2, g_1, d_1, \dots, d_{n-4}) = (1, 0, 0, 1, 0, \dots, 0)$$

$$I_0 = (k_3, k_1, k_2, g_1, d_1, \dots, d_{n-4}) = (1, 0, 0, 0, 0, \dots, 0)$$

$$\hat{T}_1 = (k_3, k_1, k_2, g_1, d_1, \dots, d_{n-4}) = (0, 0, 0, 0, 0, \dots, 0)$$

$$\hat{T}_2 = (k_3, k_1, k_2, g_1, d_1, \dots, d_{n-4}) = (0, 1, 1, 0, 1, \dots, 1)$$

$$I_1 = (k_3, k_1, k_2, g_1, d_1, \dots, d_{n-4}) = (0, 1, 1, 1, 1, \dots, 1)$$

$$\hat{T}_3 = (k_3, k_1, k_2, g_1, d_1, \dots, d_{n-4}) = (1, 1, 1, 1, 1, \dots, 1).$$

As before, the above tests consist of two ordered subsequences of tests, i.e., $\{\hat{T}_0, I_0, \hat{T}_1\}$ and $\{\hat{T}_2, I_1, \hat{T}_3\}$. Note that I_0 and I_1 are initializing sequences.

Consider the sequential network of Figure 12. Under fault-free conditions the expected output for both \hat{T}_0 and \hat{T}_1 is $(f_1, \dots, f_m) = (0, \dots, 0)$ while for \hat{T}_2 and \hat{T}_3 , the expected output is $(f_1, \dots, f_m) = (1, \dots, 1)$.

We need only show that any fault types in a loop, i.e., multiple data input faults, single faults in k_1 and k_2 and single and catastrophic faults in k_3 , will affect the output of every gate in that loop. This would imply that the fault condition would appear at the output of the loop irrespective of which gate the output comes from and hence would guarantee propagation to an output of the subnetwork S_1 . It can be shown by arguments

similar to those used in Method 1 that all other fault types will be detected by the test set defined for this method.

Now consider any distribution of data input faults in a loop in S_1 . Since a 4-T gate behaves identically to a 2-T gate with respect to data input faults, the fault closest to the loop output will propagate to this output when either \hat{T}_1 or \hat{T}_3 is applied.

It can be verified that any single fault in k_1 in a loop will cause that loop to oscillate where T_1 or T_3 is applied. A single k_2 fault causes the loop either to oscillate or set to the opposite state when the same test is applied. A single fault in the k_3 input of the 4-T gate in a loop is tested by \hat{T}_0 or \hat{T}_2 as pointed out before. A catastrophic k_3 fault appears in the output of each 4-T gate and is propagated around each loop and into other loops as data input faults and thus is detectable at an output.

Finally, catastrophic faults in k_1 and k_2 are treated in the same way as Section III.

In this section we have presented two methods of fabricating 2-T sequential networks. Each method requires 4 extra inputs and possibly an extra output. In either case we have defined a universal test set containing four tests and two initializing sequences.

V. CONCLUSIONS AND DISCUSSIONS

In this paper, we have shown that it is possible to define a logic block with two modes of operation. In the test mode, it is testable for all data input s-a-fault combinations and single faults at the control inputs with only 2 tests. In the operating mode, this logic block functions as a universal gate, capable of acting as either a NAND gate or a NOR gate depending on the logic values applied to the k_1 and k_2 inputs.

We have also shown that when combinational networks are built with this block, then these networks are testable for all single s-a-faults by a universal set of 2 tests. We have presented a method for handling the problem of catastrophic faults in a control input which may require an extra gate and output.

We have also presented two methods of fabricating sequential networks, both of which are testable for all single s-a-faults by their respective universal test sets. In one method, the problem of initializing the loops is solved by a special connection of a 2-T gate while the second method utilizes a new 4-T gate in each loop for initialization. In both these methods, we have assumed that a fault is detectable if one or more outputs are set to the opposite value or oscillate.

Note that in the methods presented, all identical control inputs are driven from a common network input. If, in the case of a combinational network, more network inputs are available, then the logic could be arranged into levels with each level having its own independent set of control inputs. In this arrangement, the first level of logic is fed from network inputs and any subsequent level is fed from preceding levels and, if necessary, network inputs while it feeds levels further down towards the output. Now, a catastrophic failure in a control input appears as a control line failure only at the level fed by that control input. To subsequent levels, it appears as distributed data input faults which are detectable. If the affected level is also the output level, then for any particular output this failure is only a single fault in a control input, which is detectable by the universal set already defined. This alternative approach, though costly, is not applicable to sequential networks since a gate can feed back on itself through a loop.

One significant feature of 2-T networks is that in the test mode logical redundancies become invisible and hence all networks are fully testable, at least for all single s-a-faults, even in the presence of such redundancies. This feature arises from the fact that in the presence of logical redundancies in a network built with conventional logic blocks, there exists at least one fault for which the value required at, at least,

one input to display the fault, i.e., try to force the opposite value from the fault, conflicts with the value required at that input to propagate the faults to an output. In a 2-T network, however, every level has either all 0's or 1's and by virtue of the special properties of the 2-T gate, any variation from that pattern at the data lines is propagated to the output.

One must note that we have considered only faults that occur at the inputs or output of a gate. We have not investigated whether there is any fault internal to the 2-T and 4-T gates which does not manifest itself as a s-a-fault at an input or output of the gate since the effect of such a fault is dependent on the specific realization of the gate on a LSI chip. We assume there is not. In defense of this assumption, it can be said that if any faults do exist that do not appear outside the gate, it may be possible either to inter-connect the internal elements of the gates in such a way that if an undetectable fault occurs, it causes a detectable fault to occur, or to use redundancy in interconnections to reduce the probability of an undetectable failure.

In defense of this assumption, it can be said that if any faults do exist that do not appear outside the gate, it may be possible either to inter-connect the internal elements of the gates in such a way that if an undetectable fault occurs, it causes a detectable fault to occur, or to use redundancy in interconnections to reduce the probability of an undetectable failure.

Finally, it must be mentioned that it is not possible in the design approach presented in this report to detect single s-a-faults within the distribution trees which drive the control inputs when such faults manifest themselves as distributed control input failures in the network. The probability of occurrence of these failures can be reduced by efficient lay-out techniques and by the use of hardware redundancy.

REFERENCES

- [1] A.D. Freidman and P.R. Memon, Fault Detection in Digital Circuits, Prentice-Hall, 1971.
- [2] H.Y. Chang, E. Manning and G. Metze, Fault Diagnosis of Digital Systems, Wiley-Interscience, 1970.

METRIC SYSTEM

BASE UNITS:

Quantity	Unit	SI Symbol	Formula
length	metre	m	...
mass	kilogram	kg	...
time	second	s	...
electric current	ampere	A	...
thermodynamic temperature	kelvin	K	...
amount of substance	mole	mol	...
luminous intensity	candela	cd	...

SUPPLEMENTARY UNITS:

plane angle	radian	rad	...
solid angle	steradian	sr	...

DERIVED UNITS:

Acceleration	metre per second squared	...	m/s
activity (of a radioactive source)	disintegration per second	...	(disintegration)/s
angular acceleration	radian per second squared	...	rad/s
angular velocity	radian per second	...	rad/s
area	square metre	...	m
density	kilogram per cubic metre	...	kg/m
electric capacitance	farad	F	A·s/V
electrical conductance	siemens	S	A/V
electric field strength	volt per metre	...	V/m
electric inductance	henry	H	V·s/A
electric potential difference	volt	V	W/A
electric resistance	ohm	...	V/A
electromotive force	volt	V	W/A
energy	joule	J	N·m
entropy	joule per kelvin	...	J/K
force	newton	N	kg·m/s
frequency	hertz	Hz	(cycle)/s
illuminance	lux	lx	lm/m
luminance	candela per square metre	...	cd/m
luminous flux	lumen	lm	cd·sr
magnetic field strength	ampere per metre	...	A/m
magnetic flux	weber	Wb	V·s
magnetic flux density	tesla	T	Wb/m
magnetomotive force	ampere	A	...
power	watt	W	J/s
pressure	pascal	Pa	N/m
quantity of electricity	coulomb	C	A·s
quantity of heat	joule	J	N·m
radiant intensity	watt per steradian	...	W/sr
specific heat	joule per kilogram-kelvin	...	J/kg·K
stress	pascal	Pa	N/m
thermal conductivity	watt per metre-kelvin	...	W/m·K
velocity	metre per second	...	m/s
viscosity, dynamic	pascal-second	...	Pa·s
viscosity, kinematic	square metre per second	...	m/s
voltage	volt	V	W/A
volume	cubic metre	...	m
wavenumber	reciprocal metre	...	(wave)/m
work	joule	J	N·m

SI PREFIXES:

Multiplication Factors	Prefix	SI Symbol
1 000 000 000 000 = 10 ¹²	tera	T
1 000 000 000 = 10 ⁹	giga	G
1 000 000 = 10 ⁶	mega	M
1 000 = 10 ³	kilo	k
100 = 10 ²	hecto*	h
10 = 10 ¹	deka*	da
0.1 = 10 ⁻¹	deci*	d
0.01 = 10 ⁻²	centi*	c
0.001 = 10 ⁻³	milli	m
0.000 001 = 10 ⁻⁶	micro	μ
0.000 000 001 = 10 ⁻⁹	nano	n
0.000 000 000 001 = 10 ⁻¹²	pico	p
0.000 000 000 000 001 = 10 ⁻¹⁵	femto	f
0.000 000 000 000 000 001 = 10 ⁻¹⁸	atto	a

* To be avoided where possible.

*MISSION
of
Rome Air Development Center*

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C³) activities, and in the C³ areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

